Asian Journal of Research in Computer Science

7(1): 50-66, 2021; Article no.AJRCOS.62354
ISSN: 2581-8260

Framework for Prioritizing Contact Tracing and Mass Testing of COVID-19 Using Graph Theory

Obed Appiah[1*], Dominic Otoo[1] and Christopher Bombie Ninfaakang[1]

[1]University of Energy and Natural Resources, Sunyani, Ghana.

Authors' contributions

This work was carried out in collaboration between all authors. Author OA conceived the presented idea. Authors OA and CBN designed the study. Author DO verified the analytical methods and supervised the findings of this work. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2021/v7i130173
Editor(s):
(1) Dr. Omidiora, Elijah Olusayo, Ladoke Akintola University of Technology (LAUTECH), Nigeria.
(2) Dr. R. Gayathri, Anna University, India.
(3) Prof. M. A. Jayaram, Siddaganga institute of Technology, India.
Reviewers:
(1) Oyeniran, Oluwashina Akinloye, Ajayi Crowther University, Nigeria.
(2) Ogundile Opeyemi Paul, Covenant University, Nigeria.
(3) Nurudeen. O. Lasisi, Federal Polytechnic Kaura Namoda, Nigeria.
(4) Bianca Constantin, Dunarea de Jos University Galati, Romania.
(5) Kamshad Mohsin, Galgotias University, India.
Complete Peer review History: http://www.sdiarticle4.com/review-history/62354

Original Research Article

Received 09 October 2020
Accepted 07 November 2020
Published 09 February 2021

ABSTRACT

Contact tracing has become one of the most useful tools for fighting the novel Corona Virus (COVID-19) pandemic worldwide. The underlining philosophy of contact tracing is determining people who have been in contact with infected persons and thus isolate them from becoming agents of onward transmission of the virus. Slow tracing of contacts and inconsistent or inaccurate information provided by patients usually leads to the spread of the virus along a trajectory at the healthcare systems' blindside. This has led to the proposal of app-based contact tracing solutions. This paper proposes an SQL-based framework that transforms simple interaction data entries into interaction graphs and applies graph theory to prioritize the contact tracing process. The framework returns nodes or individual IDs together with values called Risk_Points to enable individuals' selection for isolation and treatment. Results on simulated data show that the proposed framework can help slow the virus's rate of transmission.

**Corresponding author: E-mail: obed.appiah@uenr.edu.gh, obedkappiah@yahoo.com;*

Keywords: SQL-based framework; graph-theory risk points; contact tracing; COVID-19 testing.

1. INTRODUCTON

Contact tracing has become one of the most useful tools for fighting the novel Corona Virus (COVID-19) pandemic worldwide. The underlining philosophy of contact tracing is determining who had been near an infected person and thus isolated them, preventing them from becoming agents for onward transmission of the virus [1,2]. It is believed that the risk of infection is highest if one has been within 1.5 to 2 m of an infected person for at least 10 minutes [1]. Various works have explored contact tracing's effectiveness in dealing with the COVID-19 pandemic [1,2]. However, work by Ferretti et al. [3] shows that effective contact tracing combined with a large-scale COVID-19 testing programme might delay the spread of the virus or even stop it altogether.

The success of contact tracing in tracking and treating infectious diseases has been proven to be effective. During the outbreaks of Ebola in Africa, contact tracing was a useful tool used to track and treat patients [4-7] quickly. The main contact tracing approach is made by identifying contacts' locations through interviewing patients and their acquaintances. The information provided by the patient is verified, and their close contacts are also tested for the possibility of carrying the virus. Such close contacts are usually quarantined (isolated) until their status is known before releasing them or treating them for the virus [4–8] indicated that the dynamics of the COVID-19 transmissibility may not yet be fully understood. However, identifying patients and institutionalizing measures such as social distancing can help fight the disease. The success of contact tracing is primarily based on the accuracy of patients' information and effective means of reaching contacts. If patients' information is not consistent or inaccurate, it usually leads to the spread of the virus along a trajectory at the blindside of the healthcare system. When the contacting process is slow, tracking the virus is also slowed, especially transmission by asymptomatic patients. Inconsistencies or inaccuracies in the patient's information may not be deliberate, but recalling all close contacts within a specific period can sometimes be challenging. That is, forgetfulness and the patients' psychological state may make it difficult for them to remember all the people he/she may have come near [7]. This has led to the proposal of technology to aid in identifying proximity contacts. The smartphone-based contact tracing has been proposed for managing contacts and hence effectively identifying potential infectious people. Olu et al. [6–9] are prominent examples of proposed mobile or smartphone-based contact management systems. Various techniques or approaches such as sensitive location data (Global Positioning System or radio cell data) have been applied using mobile apps. Yasaka et al. [10] suggested: “contact points” where smartphone users create “checkpoints” by generating Quick Response (QR) codes that can be scanned by all other users when joining their checkpoint.

Abeler et al. [1] presented a slightly modified version of [10-11] concepts of app-based contact tracing. Generally, the underlining concepts are similar to the checkpoint approach, which use proximity information instead of GPS or radio cell data. The device works by users sharing their COVID-19 status on their app. Whenever a smartphone user (A) who has registered with the app on his/her phone comes near another user (B), then the IDs are exchanged, and the status is communicated if such person does have the virus or not. Fig. 1. presents an illustration of how user’s smartphones exchange IDs when phones are less than 2meters apart.

In Fig. 1, Alice and Bob stay within 2 meters for more than 10 minutes and their temporary ID is stored on each other phone. When Alice is diagnosed with COVID-19, the system can retrieve all the IDs on Alice's phone to trace all the people he came in contact with. The server then alerts all phones that have been close to the infected persons’ phone. The alerted people would still need to contact their local health authorities, as their identity is not linked to the app. Fig. 2. illustrates a pictorial representation of the concept proposed by [1].

Whether the contact tracing is done manually or app-based, there is an underlining representation of data that these interactions depict. This was identified in [12] when they proposed the use of checkpoints. Of course, the concept of checkpoints has been proposed in diverse ways such as stationary proximity (banks, restaurants, offices, etc) and mobile proximity (buses, trains, taxis, etc.). Poojary [12] computes transmission graphs of infected patients to help identify possible threats. In turn, these graphs could let every user know if any possible transmission paths were leading up to the checkpoint they

visited and thus their risk of being infected. This app's strength is that it will be able to traverse the network of contacts to effectively identify all connections to a checkpoint and estimate the risk by using the depth from an infected to a desired node. The use of networks then provides an interesting option to go beyond contact tracing. However, the interactions represented by the networks or graphs could be a starting point for going beyond the current traditional method of contact tracing to further reduce the transmission rate of the virus, especially among asymptotic patients.

The modeling of interactions of entities using graphs and trees is not new and has seen extensive applications with various degrees of success. The application of graph theory to solve various problems have underlined various models to help provide optimum solutions. The use of graphs to estimate the shortest possible paths, maximum and minimum flow, traveling salesman problem, etc. Numerous algorithms that have been proposed to fix such problems are common today [13-15]. For example, [16] applied graph theory to rank contractors successfully. In [17], graph theory was successfully used to estimate the shortest possible path that Zoomlion-Ghana garbage trucks could traverse the road network efficiently using minimum time and optimized garbage collection from various homes to dumping sites. The Contact-Tracing approach, which has become a major approach to identifying potential Coronavirus carriers, lends itself to a tree-like model with patients as nodes [12,18]. Countries worldwide have adopted such models and the World Health Organization (WHO) approves such an approach as one of the effective means of dealing with the disease [19]. Keeling [18] for instance, demonstrated how the network information could be used to perform contact tracing effectively. In their work, an illustration of graphs or trees presenting duration of contact and proximity as well is done. Fig. 3. presents a simple graph of interactions of citizens in the spread of the COVID-19 virus.

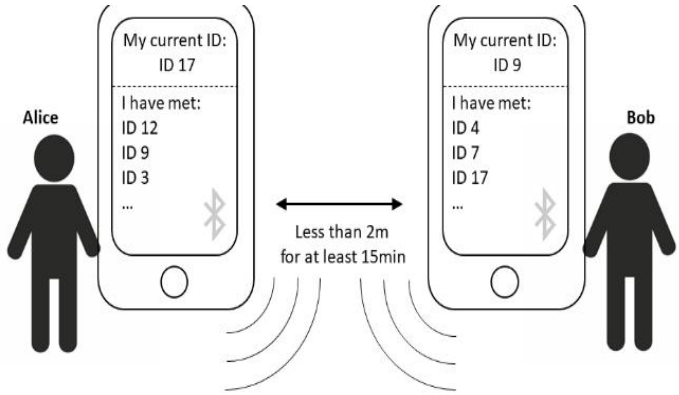

Fig. 1. A COVID-19 tracing approach via Bluetooth [Source: [1]]

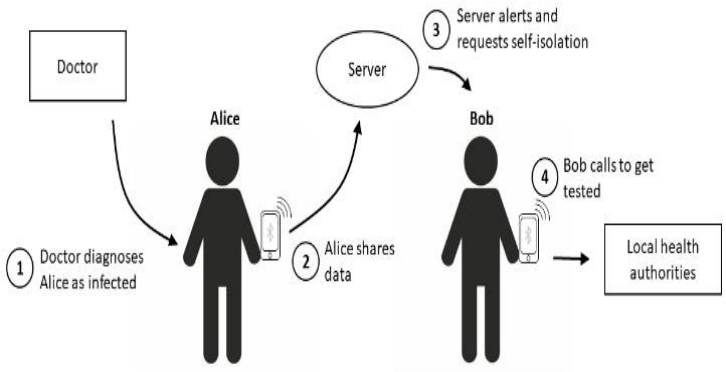

Fig. 2. A user can share their data with the server after receiving a COVID-19 diagnosis Source [1]

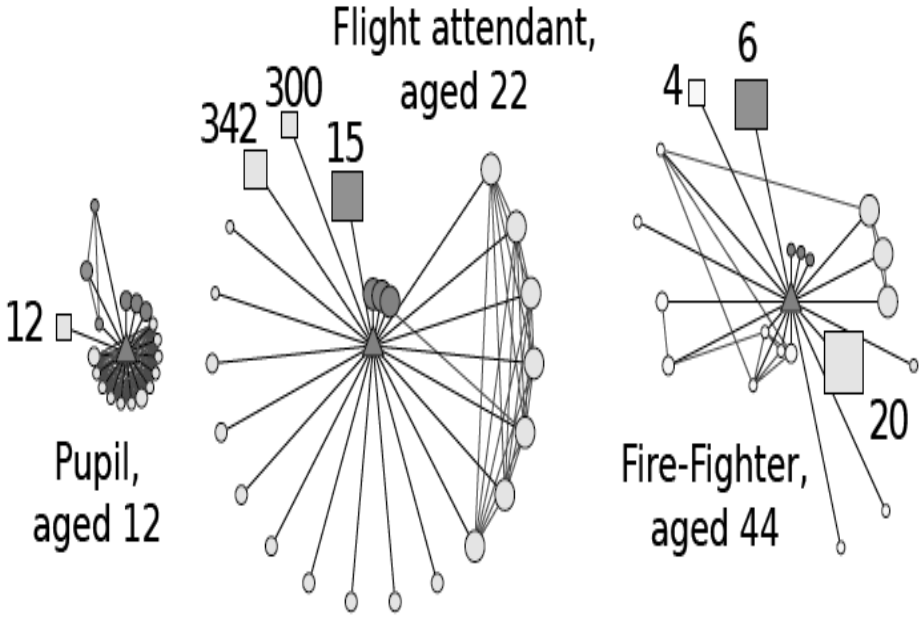

Fig. 3. A graph representation of sample contacts of members of society: Source: [18]

Contact tracing, isolation, and treatment have become an effective means of slowing the spread of COVID-19, but researchers have asked how the disease can further be slowed and that beyond contact tracing, what else can be done [20]? Mass testing has been proposed but, in the instance where resources are limited, such as in developing countries, priority testing as an alternative to mass testing may be an effective means of further slowing the spread. In this paper, a framework for generating Risk_Points ranking value is proposed for prioritizing contact tracing and mass testing to slow further the virus's infection rate given that an app-based contact tracing technologies are adopted. Again, the use of Structured Query Language (SQL) based graph data as well as manipulations to generate this ranking information is presented. The need for SQL based graph is important because most app-based applications will store their data in a relational database.

2. METHODOLOGY

The proposed framework extracts interaction information from the dataset of the contact tracing app. An interaction graph is generated from the interaction information, after which graph theory concepts such as adjacent, degrees, and paths are used to estimate a value called the Risk_Points. The Risk_Points value help identify who must be identified immediately for testing. Graph theory principles are employed in the proposed framework to prioritize contact tracing, and mass are presented next.

Fig. 4. presents a simple graph generated from the Contact table and stored in the edge's relation or table. An edge in a graph represents the connection between two nodes. In this proposed framework, personids become nodes, and a paired entry of personids in the edges table forms an edge. Therefore, the edges table store the graph representing the interaction of persons.

Adjacent (neighbours): Two vertices "a" and "b" in a graph G are called adjacent (or neighbours) in G if {a, b} is an edge of G. Adjacent or neighbours of a node in the graph represents primary contact. If a person "a" tests positive, then person "b" must be identified for testing. SQL_2 in the appendix presents an SQL statement that lists all the adjacent nodes in the interaction graph.

Degree: The degree of a vertex in an undirected graph is the number of edges incident with it, except that loop at a vertex contributes twice to the vertex's degree. Persons or nodes with a higher number of degrees are assigned higher Risk_Points. SQL 3 presents an SQL statement that identifies all nodes and their degrees

Isolated vertex: A vertex with a degree of zero (0). It follows that the isolated vertex is not adjacent to any node. Isolated nodes do not have primary contacts and therefore, the framework eliminated them in identifying contacts.

Pendant: A vertex is a pendant if and only if it has a degree of 1. In selecting persons or nodes for testing, the pendant is assigned the least Risk_Points. If pendants exist in the interactions graph, they will be listed at the bottom of the list that SQL 3 in the appendix generates. However, they can be specifically extracted using SQL 4.

In-degree and out-degree are generally related to directed graphs where arrows depict the direction of the edge. The summation of in-degree and out-degree is equal to the node's degrees and that can be estimated by SQL 3. Graphs extracted for the estimation do not employ these properties. However, edges are weighted that enables the framework to avoid retrospective traversal in the network. The weights represent the chronological order for which interaction occurs.

Regular graphs: A regular graph is a type of undirected graph where the degree for all the vertices in the graph is the same. The complete graph on vertices, denoted by K_n is the simple graph that contains exactly one edge between each pair of distinct vertices. In the situation where the interaction graph is complete, each community member is equally susceptible to the virus and therefore the RisK_Points will be the same for all members. SQL 5 in the appendix list all the nodes in the network when the graph is regular or complete.

Cuts vertices (articulated points) and cuts edges or bridges: An undirected graph is called connected if there is a path between every pair of distinct vertices of the graph. The removal of the cut vertex or cutting edges or bridges from a connected graph produces a not connected subgraph. Cuts in the interactions are assigned higher Risk_Points. SQL 6 presents an SQL statement that identifies potentials cuts in the networks.

Cycle: The cycle C_n, n >= 3, consists of n vertices v_1, v_2, ….,v_n and edges {v_1, v_2}, {v_2, v_3}, … ,{v_{n-1}, v_n} and {v_n,v_1}. When the graph is not complete, the vertices that form a cycle in the graph are assigned higher Risk_Points. A wheel W_n is obtained when we add a vertex to the cycle C_n, for n>= 3, and connect this new vertex to each of the n vertices in C_n, by the new edge.

The center of the wheel is assigned higher Risk_Points.

Path: Given two nodes or vertices, a path exists between them if a collection of edges links the vertices. Thus, a path contains the vertices $v_o,v_1,..v_k$ and edges $(v_o,v_1),(v_1,v_2),\ldots,(v_{k-1},v_k)$. A path of length *k* from a vertex *u* to a vertex *u'* in a graph $G = (V, E)$ is a sequence $\{v_0, v_1, v_2,\ldots,v_k\}$ of vertices such that $u=v_0$, $u'=v_k$, and $(v_{i-1},v_i) \in E$ for $i=1,2,\ldots,k$. The length of the path is the number of edges in the path. A path is simple if all the vertices in the path are distinct.

Path Length (k): For simplicity in the interaction graph, we assign specific names to paths of certain lengths. The path with length zero (0) will be assigned to the initial vertex. The adjacent nodes of a specific node will have a length of 1. The length can be used to describe the respective depth of contact to a particular person. The length and descriptions can be presented as follows:

Length	Description
0	Root node
1	Primary Contact
2	Secondary Contact
3	Tertiary Contact
4	Quaternary

SQL 6–8 presents various SQL statements that can select all possible paths between nodes with respective lengths of interest. The path length of 1 has already been discussed as adjacent nodes.

2.1 Generation of Graph from Persons' Interaction Dataset

The proposed framework relies on the datasets generated by app-based contact tracing tools. The framework's input is a relational database table with three (3) relevant fields storing entity IDs, location IDs, and the period a contact happened. The Unique ID for a person, Unique ID for an instance of contact (venue or location where the distance between entities happens to be less than a meter for a minimum of 10 minutes), and the period (date and time) that a Unique ID was at that specific contact instance. Table 1 represents an instance of the tblContact table in the database.

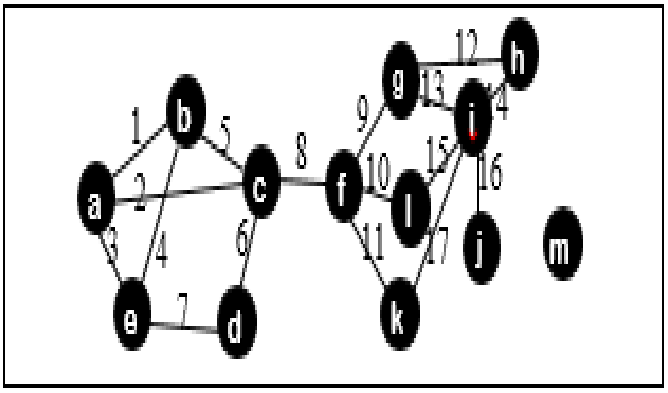

Fig. 4. A simple graph

Table 1. Contact relation with sample data

Personid	Locationid	Period
A	X	2020-04-23 18:00:00
B	X	2020-04-23 18:30:00
C	X	2020-04-23 18:10:00
D	X	2020-04-23 18:20:00
B	Y	2020-04-24 18:00:00
F	Y	2020-04-24 18:20:00
C	Z	2020-04-24 18:00:00
E	Z	2020-04-24 18:10:00

2.1.1 Algorithm for extracting graph

The interaction graph is extracted from the tblContact table by

1. Select all interactions within the last 14 days. Sort the list in ascending order.
2. Group your selection in step 1 by the locationid.
3. For each locationid, group persons whose interaction period is less than the specified minimum time for possible transmission of the disease. In this instance, 10 minutes is selected.
4. For each group identified in step 3, extract pairs of personids that meet the time frame specified.
5. Store the pair values as edges with each of the personid as nodes in the database's edges tables.

SQL 1 in the Appendix is used to generate the edges table. The edge table or relation stores all the interactions which represent the network. The chronological order of interactions is numbered and used as the weight of edges. Table 2. represents an example of entry values on the edge tables.

Table 2. An extract from edges relation

node_a	node_b	Weight
A	B	1
C	B	2

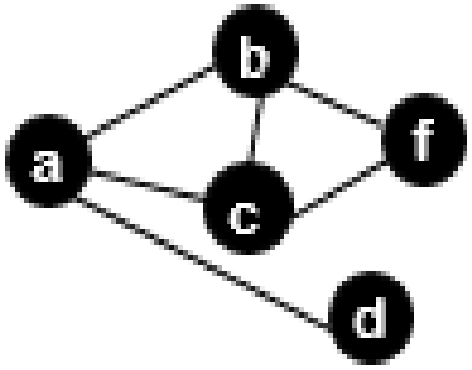

Fig. 5. A sample wheel graph

2.2 Risk_Points Estimation

The proposed framework uses a value called Risk Points to determine which person in the interaction graph must be selected for testing. The points are estimated by evaluating the degree, pendant, regular graphs, complete graph, cycle, wheels, and cuts vertices (articulated) properties of the graph. The formula is simplified by identifying all the possible paths formed up to the 4^{th} path length from a given node and then registering the second node in each path. The selected nodes' frequencies are used as Risk Points values. The main principle behind the formula is that nodes that participate a lot in interactions will have high frequencies and therefore, can be used as an estimate for reporting a vulnerability.

Algorithm 1. Prioritization primary contacts of positive COVID-19 case

0	Create AdjacentList
1	Let v_o be vertex that test positive for COVID-19
2	Extract all paths to the 4th length starting from v_0 (v_0, v_1, v_2, v_3, v_4). Use SQL 8
3	Add v_1s in all the paths in (2) to AdjacentList
4	Extract all paths to the 3rd length starting from v_0 (v_0, v_1, v_2, v_3). Use SQL7
5	Add v_1s in all the paths in (4) to Adjacent List
6	Extract all paths to the 2nd length starting from v_0 (v_0, v_1, v_2). Use SQL 7
7	Add v_1s in all the paths in (6) to AdjacentList
8	Extract all paths to the 1st length starting from v_0 (v_0, v_1). Use SQL 2
9	Add v_1s in all the paths in (8) to AdjacentList
10	Determine the frequency of vertices in AdjacentList and sort in descending order
11	Use the AdjacentList in the order of appearance to locate members of the community, with person IDs on top of the list given the highest priority.

Algorithm 2. Priority mass testing selection

- Create List
- For each entity (node), determine its adjacent nodes and their Risk_Points
- Determine the sum of Risk_Points for each adjacent node determine above
- Sort the list in descending order to identify persons with high Risk_Points.

Two algorithms for identifying members to test for COVID-19 are presented in the proposed framework. In Algorithm 1, the framework accepts a vertex, v_0, that has tested positive for COVID-19 as input and generates a list of entities and their Risk Points. The design prioritizes adjacent nodes or primary contacts by evaluating their interactions with other vertices. Priority is given to primary contact that has a lot of edges to the quaternary level from v_o. All the primary contacts will be selected for testing, but the framework ensures nodes with many secondary, tertiary, and quaternary interactions are given high Risk_Points values.

The adjacent list presented in Algorithm 1 smartly orders the primary contacts so that primary contacts with a lot of further contact interactions to the 3rd length may potentially be spreading the disease and therefore will have to be immediately tested to track the disease effectively. SQL 10 in the appendix presents the SQL statement that implements Algorithm 1.

Algorithm 2. scans through the contact interactions and uses vertices' characteristics to select nodes' whose positive testing for COVID-19 may be disastrous.

Algorithm 2 evaluates the Risk_Points values for all the nodes in the network. These values assist in prioritizing the identification and selection of community members for mass testing for COVID-19. SQL 11 in the appendix presents the SQL statement that implements Algorithm 2.

3. RESULTS AND DISCUSSION

The dataset used to test the proposed framework was generated from a simulated scenario. 10 persons, 10 unique locations, and a time interval of 5 minutes for 5 hours were generated. A set of Unique IDs {PID01, PID02, PID03, PID04, PID05, PID06, PID07, PID08, PID09, and PID10} were assigned to 10 people. A set of unique location ids {LID01, LID02, LID03, LID04, LID05, LID06, LID07, LID08, LID09, and LID10} were assigned to the various locations an individual could be found. The *neglist* table stores personids that test negative for COVID-19 to eliminate them from the extracted network or graph. Table 3. presents the capturing of persons and their location for 5 hours.

From Table 3, the time interval between two entry values is 5 minutes, but in a real-world scenario, this will change. There will be instances where the time difference may be less than a minute or more than 10 minutes. However, the 5minutes interval was selected to evaluate the proposed framework.

Table 3. Sample data on interactions in the tblcontact

Personid	Locationid	Period
PID01	LID01	20/07/2020 06:00
PID02	LID02	20/07/2020 06:05
PID03	LID03	20/07/2020 06:10
PID04	LID04	20/07/2020 06:15
PID05	LID05	20/07/2020 06:20
PID06	LID06	20/07/2020 06:25
PID07	LID07	20/07/2020 06:30
PID08	LID08	20/07/2020 06:35
PID09	LID09	20/07/2020 06:40
PID10	LID10	20/07/2020 06:45
PID09	LID02	20/07/2020 06:50
PID02	LID06	20/07/2020 06:55
PID05	LID01	20/07/2020 07:00
PID04	LID10	20/07/2020 07:05
PID08	LID04	20/07/2020 07:10
PID10	LID05	20/07/2020 07:15
PID06	LID08	20/07/2020 07:20
PID07	LID09	20/07/2020 07:25
PID09	LID04	20/07/2020 07:30
PID04	LID07	20/07/2020 07:35
PID04	LID08	20/07/2020 07:40
PID06	LID06	20/07/2020 07:45
PID03	LID02	20/07/2020 07:50
PID03	LID06	20/07/2020 07:55
PID04	LID10	20/07/2020 08:00
PID01	LID01	20/07/2020 08:05
PID04	LID03	20/07/2020 08:10
PID01	LID04	20/07/2020 08:15
PID01	LID05	20/07/2020 08:20
PID02	LID06	20/07/2020 08:25
PID07	LID07	20/07/2020 08:30
PID04	LID05	20/07/2020 08:35
PID10	LID08	20/07/2020 08:40
PID06	LID01	20/07/2020 08:45
PID05	LID01	20/07/2020 08:50
PID05	LID09	20/07/2020 08:55
PID01	LID10	20/07/2020 09:00
PID04	LID07	20/07/2020 09:05
PID02	LID03	20/07/2020 09:10
PID09	LID03	20/07/2020 09:15
PID04	LID09	20/07/2020 09:20
PID07	LID07	20/07/2020 09:25
PID02	LID08	20/07/2020 09:30
PID07	LID03	20/07/2020 09:35
PID06	LID10	20/07/2020 09:40
PID08	LID02	20/07/2020 09:45
PID05	LID04	20/07/2020 09:50
PID08	LID03	20/07/2020 09:55
PID09	LID06	20/07/2020 10:00
PID07	LID03	20/07/2020 10:05
PID01	LID10	20/07/2020 10:10
PID01	LID10	20/07/2020 10:15
PID04	LID04	20/07/2020 10:20

Personid	Locationid	Period
PID09	LID01	20/07/2020 10:25
PID08	LID09	20/07/2020 10:30
PID09	LID09	20/07/2020 10:35
PID08	LID08	20/07/2020 10:40
PID04	LID01	20/07/2020 10:45
PID02	LID09	20/07/2020 10:50
PID04	LID03	20/07/2020 10:55

Table 4. presents the set of all edges extracted from the tblContact table in Table 3. The node_a and node_b fields on the table stores the pair of nodes that form an edge. The Weight field store the chronological order that an instance of contact occurred.

48 edges were extracted from the tbl Contact table. This was done by using location and the time interval of an hour between two persons at a specific location. The graph is presented in Fig. 6. The nodes in the graph represent individuals, and the edges represent instances of contact. The weight of the graph represents the chronological order of instances of contact. In Fig. 6, an edge may have more than one weight value. This indicates that two nodes have more than an instance of interaction. An example of such 2 nodes are PID07 and PID04.

In Table 5, the vertices and their degrees are presented. This information helps in prioritizing the identification of members of the community for testing. However, the degree is not the only information used for prioritizing selection. The current contact tracing techniques being used to combat COVID19 select adjacent nodes without any priority.

The interaction graph in Fig. 6. is simplified by merging several edges into one. The value(s) attached to an edge depicts the number of edges combined to form that edge. Node PID04, has the highest number of edges, while PID10 has the least. Tables 6, 7 and 8 present the priority contact tracing list when PID07, PID04, and PID10 test positive for COVID-19, respectively. Algorithm-I was used to generate the table.

Algorithm 2 generated Table 9, which contains all the nodes in the network and their Risk_Points to help identify the community members who may be vulnerable. Table 5 shows that the PID 04 and PID10 have degrees of 15 and 4 respectively, but have 132 and 87 points on the general risk_points estimation in Table 9. These values put PID04 and PID10 on the 3rd and 6th positions, respectively for testing.

Fig. 7. presents the entities and their risk_points from Fig. 6. From the results, there is no direct relationship between the degree of nodes and their risk points.

In Table 8. we have the list of all adjacent nodes of PID04 and their Risk_Points. 8 out of the 9 nodes in the network were identified as having the potential of being infected with the disease from PID04. PID03 is the only node that has no direct contact with PID04. PID08 is assigned the highest risk value when PID04 tests positive; however, on the degree table, PID08 has a degree of 10. The value is lower than that of

Table 4. Edges extracted from the contact table

node_a	node_b	Weight
PID02	PID09	1
PID06	PID02	2
PID01	PID05	3
PID10	PID04	4
PID04	PID08	5
PID05	PID10	6
PID08	PID06	7
PID09	PID07	8
PID08	PID09	9
PID06	PID04	10

node_a	node_b	Weight
PID02	PID06	11
PID09	PID03	12
PID02	PID03	13
PID06	PID03	14
PID09	PID01	15
PID06	PID02	16
PID03	PID02	17
PID04	PID07	18
PID01	PID04	19
PID04	PID10	20
PID01	PID06	21
PID01	PID05	22
PID06	PID05	23
PID04	PID01	24
PID07	PID04	25
PID04	PID02	26
PID02	PID09	27
PID05	PID04	28
PID04	PID07	29
PID10	PID02	30
PID02	PID07	31
PID09	PID07	32
PID01	PID06	33
PID02	PID08	34
PID09	PID08	35
PID07	PID08	36
PID02	PID07	37
PID09	PID07	38
PID08	PID07	39
PID06	PID01	40
PID06	PID01	41
PID05	PID04	42
PID08	PID09	43
PID09	PID04	44
PID08	PID02	45
PID09	PID02	46
PID08	PID04	47
PID07	PID04	48

PID09, but PID09 has the least Risk Points when PID04 test positive. If degrees of nodes are the only means of prioritizing, then PID02 should have been the entity with the highest Risk_Points when PID04 test positive. PID09 has the least result because the only edge existing between the nodes (PID09 and PID04) weights 44. The 11 edges formed with PID09 has weights of (15, 12, 35, 43, 9, 8, 36, 1, 27, 46, 32). 10 out of the 11 edges do have weight values less than 44. This suggests that 10 of the contacts between PID09 had occurred before that of PID04. The other edge with the weight 46 formed by (PID09 and PID02) has fewer consequences since PID02 is an adjacent node to PID04 and therefore its risk of contracting the disease will be more probable from PID04 directly, rather than through PID09. Again, the secondary, tertiary, and quaternary path lengths generated by the algorithm are used to estimate adjacent nodes' frequencies in these paths to help generate the risk_points values in Tables 6 – 9. The framework assigns lower points to PID09 while higher points to PID08.

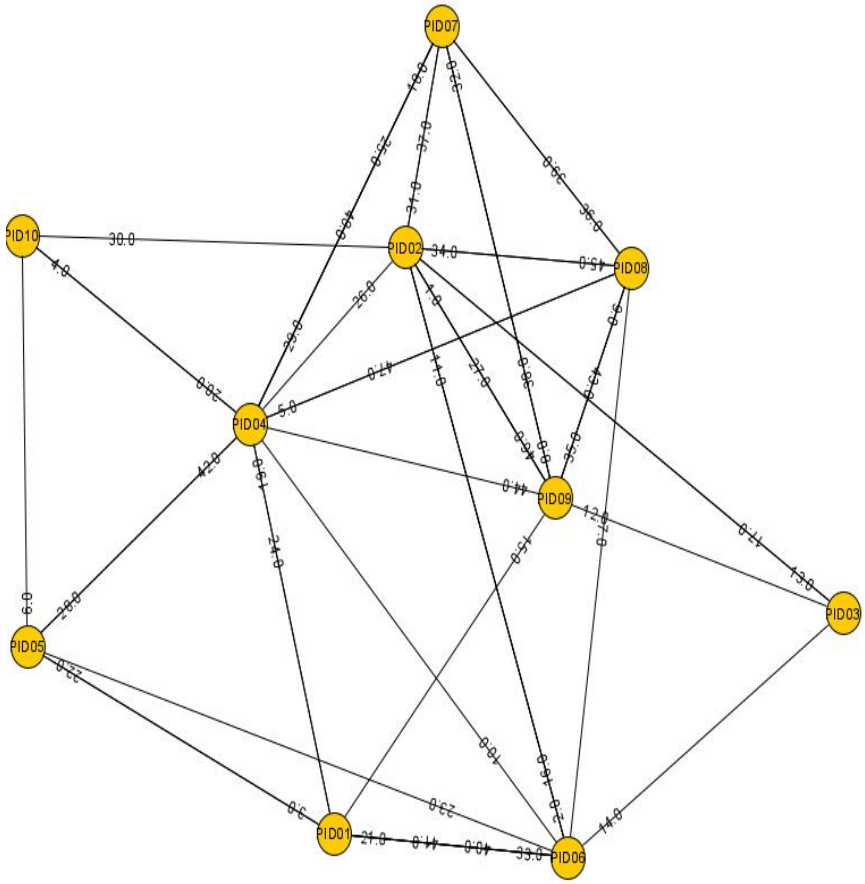

Fig. 6. Interaction network extracted from tblcontact table

Table 5. Various vertices and degrees

Vertex	Degree
PID04	15
PID02	14
PID09	12
PID06	11
PID07	11
PID08	10
PID01	9
PID05	6
PID03	4
PID10	4

Table 6. Primary contacts nodes and their risk_points when node PID 07 test positive for COVID-19

Node	Risk Points
PID04	28
PID08	9
PID09	3
PID02	2

Table 7. Primary contacts nodes and their risk_points when node PID 04 test positive for COVID-19

Node	Risk Points
PID08	48
PID07	31
PID02	31
PID10	13
PID01	5
PID05	2
PID06	1
PID09	1

Table 8. Primary contacts nodes and their risk_points when node PID10 test positive for COVID-19

Node	Risk_Points
PID04	68
PID02	18
PID05	1

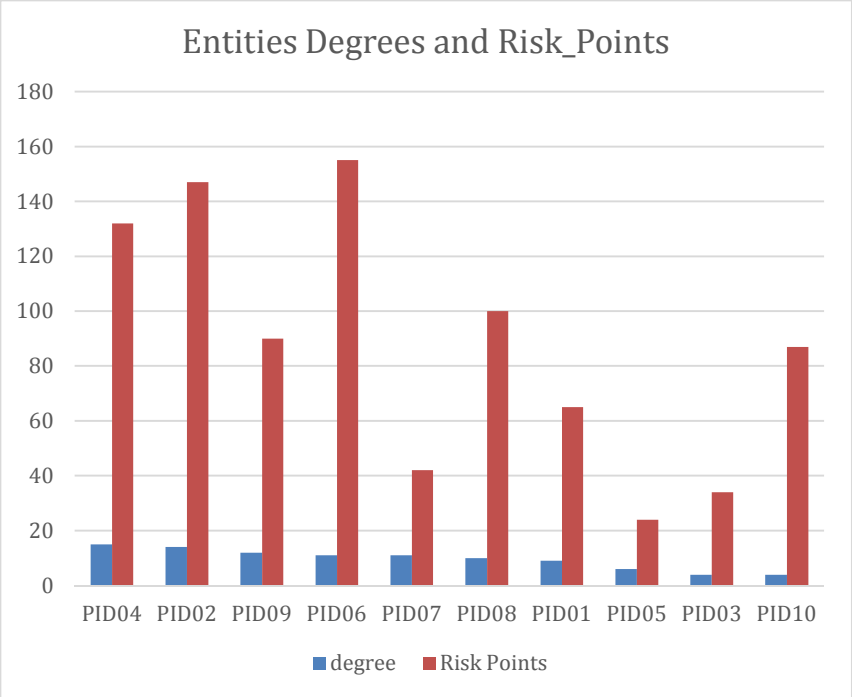

Fig. 7. Entities degrees and risk_points of the graph in Fig. 6.

Table 9. Nodes and their risk_points

Node	Risk_Points
PID06	155
PID02	147
PID04	132
PID08	100
PID09	90

Node	Risk_Points
PID10	87
PID01	65
PID07	42
PID03	34
PID05	24

4. CONCLUSION

A framework for prioritizing contact testing for COVID-19 and other infectious viruses is presented in this paper. The proposed framework relies on the interaction dataset stored by app-based contact tracing tools. It extracts interaction networks from the dataset for which edges' weights represent the chronological order of interactions. In the traditional approach of contact tracing, suspected cases do have the same alert signal. This makes it challenging to be head of the disease because the cases are not prioritized. The proposed framework assigns risk points to all contacts and can help healthcare workers decide each suspected case's urgencies and handle them accordingly. Experimental results show that the number of primary contacts does not necessarily result in high risk points. However, the time of contacts and the secondary, tertiary, and quaternary interactions influence an individual's potential of being infected with the virus. Therefore, the framework can improve the contact tracing process and hence help curb the exponential spread of the disease.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Abeler J, Bäcker M, Buermeyer U, Zillessen H. COVID-19 contact tracing and data protection can go together. JMIR mHealth and uHealth. 2020; 8(4).
2. Park O, Park YJ, Kim SY, Kim J, Lee J, Yum M. Contact transmission of COVID-19 in South Korea: Novel investigation techniques for tracing contacts. Osong Public Health and Research Perspectives. 2020;11(1):60–63,
3. Ferretti L, Wymant C, Kendall M, Zhao L, Nurtay A, Abeler-Dorner L, Parker M, Bonsall DG, and C. Fraser. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing; 2020.
4. Saurabh S, Prateek S. Role of contact tracing in containing the 2014 Ebola outbreak: A review. African Health Sciences. 2017;17(1):225.
5. Martínez MJ, Salim AM, Hurtado JC, Kilgore PE. Ebola virus infection: Overview and update on prevention and treatment. Infectious Diseases and Therapy, 2015; 4(4):365–390.
6. Olu OO, Lamunu M, Nanyunja M, Dafae F, Samba T, Sempiira N et al. Contact tracing during an outbreak of ebola virus disease in the western area districts of Sierra Leone: Lessons for future ebola outbreak response. Frontiers in Public Health. 2016;4.
7. Danquah LO, Hasham N, Macfarlane M, Conteh FE, Momoh F, Tedesco AA et al. Use of a mobile application for Ebola contact tracing and monitoring in northern Sierra Leone: A proof-of-concept study. BMC Infectious Diseases. 2019;19(1).
8. Cheng HY, Jian SW, Liu DP, Ng TC, Huang WT, Lin HH. Contact tracing assessment of COVID-19 transmission dynamics in taiwan and risk at different exposure periods before and after symptom onset. JAMA Internal Medicine; 2020.
9. He Z. What further should be done to control COVID-19 outbreaks in addition to cases isolation and contact tracing measures? BMC Medicine. 2020;18(1).
10. Yasaka TM. Lehrich BM, Sahyouni R. Peer-to-peer contact tracing: Development of a privacy-preserving smartphone app (Preprint); 2020.
11. Trace together.
    Available:http://www.tracetogether.gov.sg/.
    Accessed: 22-Jun-2020.
12. Poojary T. Coronavirus: Apple and google to partner for contact tracing technology; 2020.
    Available:https://yourstory.com/2020/04/coronavirus-apple-google-partner-contact-tracing.
    Accessed: 22-Jun-2020.

13. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. MIT Press. Cambridge. MA. 2nd edition; 2001.
14. Bryant V, Wallis WD. A beginner's guide to graph theory, the mathematical gazette. 2001;85(503):374.
15. SSG, Vetrivel S, ENM. Applications of graph theory in computer science an overview. International Journal of Engin eering Science and Technology. 2010; 2(9):4610–4621.
16. Darvish M, Yasaei M, Saeedi A. Application of the graph theory and matrix methods to contractor ranking. International Journal of Project Manage ment. 2009;27(6):610–619.
17. Otoo D, Amponsah SK, Sebil C. Capacitated clustering and collection of solid waste in Kwadaso estate, Kumasi. Journal of Asian Scientific Research, 2014; 4(8):460–472.
18. Keeling MJ, Hollingsworth TD, Read JM. The efficacy of contact tracing for the containment of the 2019 novel corona virus (COVID-19); 2020.
19. Coronavirus disease (COVID-19) - Events as they happen. World Health Organi zation.
Available:https://www.who.int/emergencies /diseases/novel-coronavirus-2019/events-as-they-happen.
Accessed: 22-Jun-2020.
20. Dropkin G. Covid-19: Contact tracing requires ending the hostile environment. Bmj p. m1320; 2020.

APPENDIX

SQL 1. Generate edges from tbl contact

```
Delete from edges;
Alter Table edges AUTO_INCREMENT = 1;
Insert into edges (node_a, node_b)selecta.personid as node_a, b.personid as node_b from contact as a, contact as b where a.locationid = b.locationid and timestampdiff(MINUTE, a.period, b.period) between 0 and 120 and a.personid != b.personid and timestampdiff(DAY, a.period, CURRENT_DATE) <= 14 and timestampdiff(DAY, b.period, CURRENT_DATE) <= 14
Delete from edges;
Alter Table edges AUTO_INCREMENT = 1;
Insert into edges (node_a, node_b)selecta.personid as node_a, b.personid as node_b from contact as a, contact as b where a.locationid = b.locationid and timestampdiff(MINUTE, a.period, b.period) between 0 and 120 and a.personid != b.personid and timestampdiff(DAY, a.period, CURRENT_DATE) <= 14 and timestampdiff (DAY, b.period, CURRENT_DATE) <= 14 and a.personid not in (select * from neglist) and b.personid not in (Select * from neglist);
```

SQL 1. Generate edges from tbl contact

```
Delete from edges;
Alter Table edges AUTO_INCREMENT = 1;
Insert into edges (node_a, node_b)selecta.personid as node_a, b.personid as node_b from contact as a, contact as b where a.locationid = b.locationid and timestampdiff(MINUTE, a.period, b.period) between 0 and 60 and a.personid != b.personid and timestampdiff(DAY, a.period, CURRENT_DATE) <= 14 and timestampdiff(DAY, b.period, CURRENT_DATE) <= 14;
Delete from edges;
Alter Table edges AUTO_INCREMENT = 1;
Insert into edges (node_a, node_b)selecta.personid as node_a, b.personid as node_b from contact as a, contact as b where a.locationid = b.locationid and timestampdiff(MINUTE, a.period, b.period) between 0 and 60 and a.personid != b.personid and timestampdiff(DAY, a.period, CURRENT_DATE) <= 14 and timestampdiff(DAY, b.period, CURRENT_DATE) <= 14 and a.personid not in (select * from neglist) and b.personid not in (Select * from neglist);
```

SQL 2. Generate adjacent nodes of a particular node

```
• Select distinct node_b as adjacent from edges where node_a = 'A' union Select distinct node_a as Adjacent from edges where node_b = 'A'
```

SQL 3. Generate degrees for each vertex in the graph

```
• Select vertex, sum(degree) as degree from (select distinct node_a as vertex, count(node_a) as degree from edges group by node_a UNION ALL select distinct node_b as vertex, count(node_b) as degree from edges group by node_b) As Deg group by vertex order by degree desc
```

SQL 4. Select pendant nodes

```
• select * from (Select vertex, sum(degree) as degree from (select distinct node_a as vertex, count(node_a) as degree from edges group by node_a UNION ALL select distinct node_b as vertex, count(node_b) as degree from edges group by node_b) As Deg group by vertex) as Pendant where Pendant.degree = 1
```

SQL 5. Regular graphs

```
Select * from
(Select vertex, sum(degree) as degrees from (select distinct node_a as vertex, count(node_a) as degree from edges group by node_a UNION ALL select distinct node_b as vertex, count(node_b) as degree from edges group by node_b) As Deg group by vertex order by degree desc) As tblDegrees
where degrees = (select max(degrees) from (Select vertex, sum(degree) as degrees from (select distinct node_a as vertex, count(node_a) as degree from edges group by node_a UNION ALL select distinct node_b as vertex, count(node_b) as degree from edges group by node_b) As Deg group by vertex order by degree desc) As tblDegrees)
AND
degrees = (select min(degrees) from (Select vertex, sum(degree) as degrees from (select distinct node_a as vertex, count(node_a) as degree from edges group by node_a UNION ALL select distinct node_b as vertex, count(node_b) as degree from edges group by node_b) As Deg group by vertex order by degree desc) As tblDegrees)
```

SQL 6. Secondary contact (path length of 2)

```
• select a.node_a as Suspected, a.node_b as Primary_Contact, b.node_bSecondary_Contact from edges as a, edges as b where a.node_b = b.node_a and a.weight<b.weight order by Suspected
```

SQL 7. Tertiary contact (path length of 3)

```
• select a.node_a as Suspected, a.node_b as Primary_Contact, b.node_bSecondary_Contact, c.node_b as Tertiary_Contact from edges as a, edges as b, edges as c where a.node_b = b.node_a and b.node_b=c.node_a and a.Weight<b.Weight and b.weight<c.Weight
```

SQL 8. Quaternary contact (path length of 4)

```
• select a.node_a as Suspected, a.node_b as Primary_Contact, b.node_bSecondary_Contact, c.node_b as Tertiary_Contact, d.node_b as Quaternary_Contact from edges as a, edges as b, edges as c, edges as d where a.node_b = b.node_a and b.node_b=c.node_a and c.node_b = d.node_a and a.Weight<b.Weight and b.weight<c.Weight and c.Weight<d.Weight
```

SQL 9. Identify potential cuts

```
Select Primary_Contact, Secondary_Contact, count(Tertiary_Contact) as Freq from (select a.node_a as Suspected, a.node_b as Primary_Contact, b.node_bSecondary_Contact, c.node_b as Tertiary_Contact from edge as a, edge as b, edge as c where a.node_b = b.node_a and b.node_b=c.node_a order by Suspected, Primary_Contact, Secondary_Contact, Tertiary_Contact) As G group by Primary_Contact, Secondary_Contact order by Freq DESC
```

SQL 10. SQL implementation of algorithm 1

```
Select AL4 as Node, count(AL4) as Risk_Points from
((select a.node_b as AL4 from edges as a, edges as b, edges as c, edges as d where a.node_b = b.node_a and b.node_b=c.node_a and c.node_b = d.node_a and a.Weight<b.Weight and b.weight<c.Weight and c.Weight<d.Weight and a.node_a = 'A')
UNION ALL
(select a.node_b as AL3 from edges as a, edges as b, edges as c where a.node_b = b.node_a and b.node_b=c.node_a and a.Weight<b.Weight and b.weight<c.Weight and a.node_a = 'A')
UNION ALL
(select a.node_b as AL2 from edges as a, edges as b where a.node_b = b.node_a and a.Weight<b.Weight and a.node_a = 'A')
UNION ALL
```

```
(Select node_b as AL1 from edges where node_a = 'A')
UNION ALL
(Select node_a as AL0 from edges where node_b = 'A')) As Paths
group by AL4 order by Risk_PointsDesc
```

SQL 11. SQL implementation of algorithm 2

```
Select AL4 as Node, count(AL4) as Risk_Points from
((select a.node_a as AL4 from edges as a, edges as b, edges as c, edges as d where a.node_b =
b.node_a and b.node_b=c.node_a and c.node_b = d.node_a and a.Weight<b.Weight and
b.weight<c.Weight and c.Weight<d.Weight)
UNION ALL
(select a.node_a as AL3 from edges as a, edges as b, edges as c where a.node_b = b.node_a and
b.node_b=c.node_a and a.Weight<b.Weight and b.weight<c.Weight)
UNION ALL
(select a.node_a as AL2 from edges as a, edges as b where a.node_b = b.node_a and
a.Weight<b.Weight)
UNION ALL
(Select node_b as AL1 from edges)
UNION ALL
(Select node_a as AL0 from edges)) As Paths
group by AL4 order by Risk_PointsDesc
```


Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sdiarticle4.com/review-history/62354